Experimental Evaluation of Topology Control and Synchronization for In-Building Sensor Network Applications

W. Steven Conner Jasmeet Chhabra Mark Yarvis Lakshman Krishnamurthy
Intel Research & Development
2111 N.E. 25th Avenue
Hillsboro OR, 97124
{w.steven.conner, jasmeet.chhabra, mark.d.yarvis, lakshman.krishnamurthy}@intel.com

August 2003

ABSTRACT

While multi-hop networks consisting of 100s or 1000s of inexpensive embedded sensors are emerging as a means of mining data from the environment, inadequate network lifetime remains a major impediment to real-world deployment. This paper describes several applications deployed throughout our building that monitor conference room occupancy and environmental statistics and provide access to room reservation status. Because it is often infeasible to locate sensors and display devices near power outlets, we designed two protocols that allow energy conservation in a large class of sensor network applications. The first protocol, Relay Organization (ReOrg), is a topology control protocol which systematically shifts the network's routing burden to energy-rich nodes, exploiting heterogeneity. The second protocol, Relay Synchronization (ReSync), is a MAC protocol that extends network lifetime by allowing nodes to sleep most of the time, yet wake to receive packets. When combined, ReOrg and ReSync lower the duty cycle of the nodes, extending network lifetime. To our knowledge, this paper presents the first experimental testbed evaluation of energy-aware topology control integrated with energy-saving synchronization. Using a 54-node testbed, we demonstrate an 82-92% reduction in energy consumption, depending on traffic load. By rotating the burden of routing, our protocols can extend network lifetime by 5-10 times. Finally, we demonstrate that a small number of wall-powered nodes can significantly improve the lifetime of a battery-powered network.

1. INTRODUCTION

Networked sensors are creating a new class of applications. Many of these applications connect users with the real world to help conduct scientific field studies [16], raise business bottom lines [26], and improve everyday life [7]. These networks will consist of 100s or 1000s of networked sensors and are enabled by small, inexpensive devices such as RFID tags, smart cards, and sensors. In the not so distant future these devices will be sufficiently small and inexpensive to be embedded into the environment and networked using low-power radios. Sensor networks will be deployed in an ad hoc manner and use multi-hop networking protocols to ensure full connectivity, fault tolerance, and long operational life, making real-world information and control ubiquitously available [25].

We have developed experimental applications to solve everyday problems in our work environment that monitor occupancy of conference rooms, provide conference room status in a ubiquitous manner, and monitor environmental parameters such as temperature throughout our building [7]. These are illustrative examples from the class of in-building sensing applications for which one of the biggest hurdles towards deployment is energy provisioning. Even in buildings, where power outlets are relatively abundant, it is not always feasible to locate sensors near these outlets. While a few nodes may be wall powered, many require battery power since running new wires to sensors is expensive and time consuming. This class of applications requires self-configuring energy-efficient protocols that allow the network to survive many months, or even years, when a large number of nodes operate on constrained battery power. Nodes should be able to spend significant portions of their time sleeping to save energy, yet still maintain communication [27] [32].

This paper makes the following contributions. We specify requirements for in-building applications based on their characteristics and real-world deployment challenges. We present the design of a topology control protocol, Relay Organization (ReOrg), which systematically exploits heterogeneous nodes (wall powered/battery powered) in the context of real-world applications. ReOrg creates a network backbone reducing the number of nodes that participate in network routing. We then describe the design of Relay Synchronization (ReSync), a MAC protocol that allows nodes to sleep when not communicating and tunes the communication duty cycle between individual neighbors based on the backbone created by ReOrg. We present the elements required to evaluate the performance of energy conservation protocols in a sensor network testbed, including methods to estimate energy consumption, weigh the impact of various hardware design choices, and fairly evaluate in the presence of packet loss. We demonstrate these elements in the context of the ReOrg and ReSync protocols in a 54 node testbed of Berkeley motes [12]. To the best of our knowledge, this paper provides the first experimental testbed study of integrated topology control and synchronization protocols in a sensor network. Our results confirm previous simulation studies and provide additional insights.

The experimental evaluation is summarized by several key results. First, while a topology control protocol increases network overhead, a better network topology can reduce overall energy consumption across the network if the MAC protocol allows nodes to sleep. We demonstrate a savings of as much as 63% with existing hardware and 83% with improved hardware. Second, by tightly integrating backbone creation with the MAC protocol, we achieved a reduction in energy consumption of between 82-92%. A benefit is still obtained after considering packet loss. We further demonstrate that because our protocols allow nodes to take turns sharing the burden of packet forwarding, this 82-92% gain can be materialized into a 5-10 fold increase in network lifetime. Thus, individual nodes are not necessarily sacrificed to achieve an overall energy reduction. Finally, we demonstrate that our protocols take advantage of hetero-

* Other names and brands may be claimed as the property of others.

geneity to further increase network lifetime. As a result, the lifetime of a battery-powered network can be increased through the addition of a small number of wall-powered nodes.

2. IN-BUILDING APPLICATIONS

In-building applications form a large class of sensor network applications. We have deployed several experimental applications in our workplace that perform a variety of functions such as monitoring conference room occupancy, providing conference room occupancy and reservation status in a ubiquitous manner, and monitoring environmental parameters such as temperature. In this section we provide application details to motivate the network protocol design.

In many modern office complexes, closed-wall offices have been replaced with high-density cubicles to inspire an atmosphere of open collaboration and accessibility among people in the building. However, the lack of private offices means that meetings must be held in conference rooms. Since most rooms are reserved days or weeks in advance, it is often not possible to reserve a room with little or no notice. However, meetings commonly do not last the entire reservation time, and it is common for users to wander around a building in search of an empty room for an impromptu meeting. We have built a system that provides people access to room occupancy status and allows them to find empty rooms from handhelds and PCs.

Our system consists of a network of sensors deployed in and around conference rooms. In-room sensors are connected to motion detectors (Figure 1 (a)), which monitor room occupancy status. A gateway node receives the sensor data which is aggregated and stored to provide status information to desktop users over the web. Figure 2 (a) shows a screen shot from a web application that provides live occupancy information for rooms on a given building floor. Users of this application can avoid searching for a conference room and walk directly to an empty room. We have also connected PDAs to the sensor network, allowing mobile users to obtain the status of nearby conference rooms directly (Figure 2 (b)). Occupancy information is also available via status nodes at the end of the aisles that indicate the presence of an empty room in that aisle.

In addition to providing live occupancy data, motion detector data may also be compiled over time for future analysis. Figure 2 (c) illustrates an application that compares gathered room usage statistics with data from the online reservation system. Such data allows the automatic identification of individuals who consistently reschedule meetings without canceling room reservations and allows facility analysts to analyze building usage patterns. Typical conference room usage patterns can aid the design of new buildings. These usage scenarios require real-world information to be gathered, processed, stored and made available in a ubiquitous manner. In addition to these applications, the infrastructure has been reused to gather temperature and battery voltage/current usage at each node.

Our application also makes room reservation status, normally available through an Outlook® reservation service, available at each room. Status nodes at the entrance to each conference room indicate current/future reservation status (Figure 1 (b)). This status information is pushed from the gateway to individual nodes, using the reverse path generated by the data collection tree. To save battery power these nodes can turn on when a user presses the status button, prolonging the life of the node. Future versions of this node will include an inexpensive display to indicate room ownership.

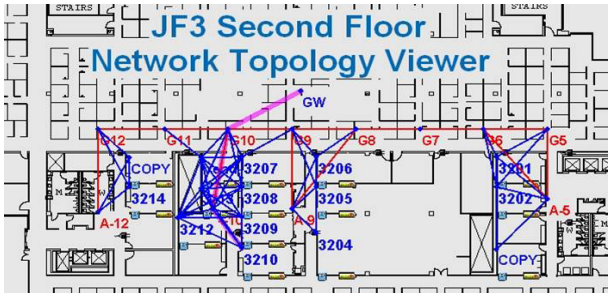
(a) (b)

Figure 1: Conference room (a) motion sensor node and (b) reservation status indicator.

This service helps resolve room reservation conflicts that are settled today through phone calls to conference room administrators.

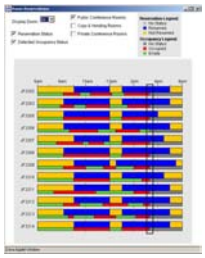
Power was the most significant challenge we faced when installing these applications. To simplify the deployment of our application and to avoid installation of new wires, we used battery powered devices, allowing an ad hoc network deployment at minimal cost. The power issue is not limited to the conference room application. In our analysis of building- and factory-monitoring applications, eliminating both power and network wiring at the sensors results in a significant cost savings and improved return on investment.

These deployment and maintenance challenges motivate the design of our protocols. While we leverage power outlets available throughout the building, many of the sensors and devices are battery powered. One of our primary objectives is to take advantage of wall-powered nodes to yield energy savings for battery powered nodes. However, all battery powered nodes will not be within one hop of wall-powered nodes. A topology control protocol is required to leverage heterogeneity and a synchronization protocol is required to allow battery powered nodes to sleep yet still communicate.



(a)

(b)



(c)

Figure 2: Applications. (a) Web page showing live conference room occupancy and network topology in the building. (b) PDA application showing status of rooms in the vicinity of the mobile user. (c) Occupancy history application comparing actual usage data to room reservations.

3. RELATED WORK

A number of approaches to save energy have been described in the literature. Network topology organization protocols have been reported as far back as 1981 [1]. These protocols may be used to reduce energy consumption by limiting the number of active nodes and links. Unlike ReOrg, most existing topology control protocols are targeted at higher-end devices without the computing and memory resource constraints of tiny sensor nodes. The notion that turning off the radio component can have a large impact on overall energy consumption is not new. Several MAC protocols allow a node to opportunistically shut off its radio and sleep. ReSync differs from existing protocols by using a distributed TDMA-like approach that avoids the negotiation overhead required to determine a global schedule and utilizes local interactions without requiring global time synchronization. In addition, several energy-aware routing protocols have been described. Rather than directly comparing the individual performance of ReOrg and ReSync to similar protocols, we focus on tightly integrating topology control and a synchronizing MAC protocol to reduce the rate at which nodes must listen to their neighbors, even for route maintenance messages. Finally, while most existing evaluations of energy saving protocols have focused on simulation, a major contribution of this paper is an experimental testbed evaluation that validates real-world benefit.

3.1 Topology Control Protocols

Topology control protocols that elect a backbone for connectivity typically implement a well-connected flat network topology to reduce the number of nodes that participate in data delivery, while still maintaining many alternate paths across the network. ReOrg belongs to this class of protocols. Baker and Ephremides [1] proposed a linked cluster algorithm to create multi-hop packet radio networks using a two-phase election of cluster heads and gateways. While similar to the link cluster algorithm, ReOrg was developed independently and has several unique features. ReOrg was designed with energy conservation as its key goal and considers the energy differences between nodes in cluster formation. Moreover, ReOrg does not require the maximum network size to be known a priori, and was designed for the limited resources of sensor networks, requiring only $O(n)$ storage in a network with an average neighborhood size of n (see Section 4.2).

Span [5] is a backbone topology election protocol that attempts to minimize the number of "coordinators" while minimizing loss of capacity or increased latency. Span elects coordinators when two neighbors of a non-coordinator are not connected via one or two coordinators. Thus nodes must exchange complete neighbor lists and store each neighbor's complete neighbor list, requiring $O(n^2)$ storage complexity. Wu *et al.* [28] describe a distributed algorithm for computing an energy-aware connected dominating set for routing in ad hoc networks which is similar to Span and also requires $O(n^2)$ storage. Compared with ReOrg, these protocols trade storage complexity for a potentially more optimal backbone.

Still other protocols allow nodes to sleep based on node redundancy in dense ad hoc networks. GAF [30] divides nodes into a grid based on their geographic positions. Nodes within each grid are considered to be equivalent, allowing all but one to sleep. Unlike ReOrg, GAF relies on geographic information, which is not always available and may not correspond directly with RF propagation, CEC [29] identifies equivalent nodes using a clustering approach similar to ReOrg. While CEC uses clustering to allow redundant nodes to sleep, ReOrg uses clustering to eliminate redundant links, reducing the number of neighbors a node must listen to. Rather than identifying topologically equivalent nodes, ASCENT [3] allows nodes to randomly choose to wake and sleep based on local density. While each node is only required to store a list of neighbors, the protocol assumes very high density and does not guarantee connectivity. All of these protocols aim to identify nodes that need not participate and are appropriate for very high density networks. ReOrg is appropriate in networks in which all nodes must sense and report values.

3.2 MAC Protocols that Conserve Energy

Several MAC protocols allow a node to opportunistically turn off its radio and sleep. Both PAMAS [24] and Sensor-MAC (S-MAC) [32] turn off the radio when a node determines that it cannot transmit or receive packets (e.g., because neighbors are transmitting or receiving). Woo and Culler's transmission control scheme [27] demonstrates energy savings by reducing unnecessary listening while contending to acquire the channel. These protocols conserve energy by turning off the radio (and potentially the processor) when it is not able to send or receive.

Even more energy can be saved by energizing a node only when it is scheduled to communicate, effectively reducing the duty cycle. Time-division multiple-access (TDMA) can be used to save energy by allowing nodes to sleep when they are not scheduled to communicate. Commonly used in wireless LANs [14] and single-hop sensor networks [11], TDMA schemes typically require a centralized controller to schedule communication. ReSync distributes the synchronization function, eliminating the need for an explicit "master" node. The linked cluster algorithm [1] also distributes the scheduling task among elected cluster heads throughout the network, but it assumes global clock synchronization and priori knowledge of the network size. ReSync provides local, rather than global, synchronization, allowing nodes to synchronize only with their neighbors. Other TDMA-like protocols also overcome these problems. SMACS [25] is a hybrid TDMA and frequency-division multiple-access (FDMA) protocol in which pairs of nodes independently select communication time slots. Rather than avoiding TDMA collisions, SMACS uses frequency division to allow a node to communicate with each neighbor. This approach requires sensor network radios to be capable of operating on multiple frequencies. ReSync approximates TDMA properties without requiring centralized coordination, global time synchronization, or multiple frequency channels.

Other protocols establish a global wake/sleep schedule and use contention to communicate. 802.11 ad hoc power saving mode [14] uses a periodic beacon, followed by an ATIM window during which nodes advertise traffic for the given beacon interval. Nodes with no traffic to send or receive may sleep until the next beacon. This technique was designed for a single-hop communication cell, but similar approaches, such as Sensor-MAC (S-MAC) [32], extended it to multi-hop networks. S-MAC establishes neighborhood sleep/wakeup schedules. While many nodes typically share the same schedule, each node exchanges its schedule with its neighbors and must remain awake during the union of its own schedule and the schedules of unsynchronized neighbors. While both S-MAC and ReSync allow a node to reduce its duty cycle through local neighborhood synchronization, S-MAC is contention-oriented while ReSync schedules use of the channel.

Despite a few exceptions (for example [27] [32]), the body of research to evaluate the energy performance of MAC protocols for

sensor networks to date has relied largely on simulations. This paper, on the other hand, presents an experimental testbed study.

3.3 Energy Aware Routing

Several protocols have been designed to account for energy when choosing routes in an ad hoc network. Power-Aware Routing [24] uses routing metrics based on the available energy of each node. Minimum Energy Routing [8] reduces energy consumption by choosing routes that consume minimal energy, assuming the ability to select the minimal transmit power required for communicating with neighbors. Energy Conserving Routing [4] uses a similar approach and rotates routes to distribute energy consumption across the network. While these protocols could supplement the energy conservation achieved by backbone protocols such as ReOrg, without a backbone these protocols will consume a great deal of excess energy for route discovery and maintenance. GEAR [33] eliminates flooding by using a heuristic based on both geographical information and energy awareness to route packets to the geographic region of the destination node. When geographic information is available and the geographic location of the destination node is known, this approach can greatly reduce the cost of route discovery.

3.4 Layer Integration

Previous studies have shown that protocol integration and interaction have a significant performance impact. In the ad hoc networking space, such studies have typically focused on interactions between routing and MAC protocols [2] [22]. Span [5] has been integrated with 802.11 power saving mode, allowing non-coordinators to sleep and save energy. Integration of GAF and STEM [23] was shown to enhance energy savings by increasing sleep time relative to node equivalence and latency. This paper describes a similar strategy of integrating protocols to increase energy conservation.

4. REORG: BACKBONE CREATION

The primary goal of ReOrg is to conserve energy and extend the lifetime of a network by reducing the number of nodes that participate in routing. Preference in router selection is given to nodes with more available energy, allowing ReOrg to exploit wall-powered nodes. A mixture of battery and wall-powered nodes is typical of indoor sensor network applications. When nodes with similar energy capacities must participate in routing, they take turns sharing the burden of forwarding packets. Figure 2 (a) illustrates a backbone elected by ReOrg in a typical office environment. This example illustrates a conference room monitoring sensor network (Section 2). Out of the 30 sensor nodes deployed in this irregular topology, ReOrg elected a backbone of 13 relays to maintain full network connectivity while allowing nodes to sleep.

ReOrg generates and maintains a backbone to route messages between any pair of nodes in a multi-hop network, allowing non-backbone nodes to spend most of their time sleeping. Section 5 describes ReSync, a MAC protocol that can be combined with ReOrg to allow nodes to sleep while participating in routing. The design of ReOrg is motivated by the following observations. First, in many deployments, sensor nodes can have different amounts of available energy, ranging from wall power to small capacity batteries, and are not equal candidates for relaying packets. Finally, in a dense network, not every node is required to relay messages to achieve a fully connected network [30]. ReOrg self-organizes an adaptive backbone of relays, consisting of a subset of nodes in the network, over which data will flow between sources and destinations. As shown in Figure 3, ReOrg does not attempt to generate a spanning tree or other loop-free topology. Rather, a fully-connected graph (as long as one exists in the network) is generated to provide many alternate paths across the network. Routing protocols running at a layer above ReOrg can be used to identify loop-free routes.

Reducing the number of nodes involved in multi-hop communication has several benefits. First, non-backbone nodes can potentially sleep most of the time since they do not have to be awake to forward messages for their neighbors. Moreover, the rotation of roles performed by nodes has been shown to conserve overall energy usage across a network [17]. Second, it reduces the effect of higher-layer flooding, which is the fundamental mechanism for route maintenance in most ad hoc routing protocols such as directed diffusion [13] and AODV [19]. Reducing the number of nodes that participate in flooding has been shown to reduce contention and collisions in dense networks [18]. Third, if nodes must listen to fewer neighbors to forward messages, they can spend more time sleeping.

A relay backbone exhibits the graph theoretical properties of a connected dominating set. Computing a minimum connected dominating set is NP-complete [6], thus finding the minimum number of nodes required to guarantee connectivity in a moderate sized network is not realistically feasible. However, achieving a minimal set is not required to attain energy savings (as will be demonstrated in Section 6). Instead, the design of ReOrg focuses on achieving low storage, communication, and computational complexity to allow implementation on resource-constrained sensor devices.

4.1 Description of ReOrg Protocol

ReOrg uses a distributed approach to generate a relay backbone. First, nodes select primary relays such that every node is either a primary relay or has one as a neighbor. Primary relays are chosen based on a normalized metric (e.g., remaining energy capacity, remaining time to live, computing resources, or neighbor count) which rates a node's suitability as a router. To guarantee connectivity across the network, gaps between selected primary relays must then be filled. Non-relay nodes select themselves as secondary relays based on their ability to bridge such gaps, with higher relay metric nodes taking priority.

ReOrg uses a two-phase algorithm based on the exchange of periodic *OrgHello* messages between neighbors. In the first phase, nodes identify their immediate neighbors and select the neighbor with the best relay metric as a primary relay. When selecting primary relays, nodes only consider neighbors to which they have links meeting a minimum quality threshold (e.g., > 80% success rate). Nodes inform their chosen primary relay using a field in the *OrgHello* message. After this phase, each non-relay node is within one hop of at least one primary relay (the one it selected). Once primary relay selection has occurred, gaps in connectivity of at most

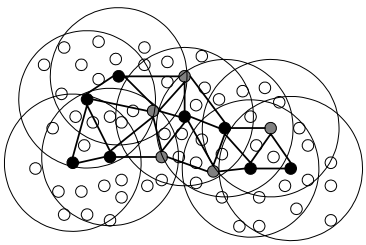


Figure 3: Example relay backbone selected by ReOrg in a 70 node network, requiring 19% of nodes to be relays.

three hops may remain between primary relays (as shown in Figure 4). An additional step is required to ensure a fully connected graph.

In the second phase, each node identifies relays in its two-hop neighborhood and learns whether these primary relays are connected, either directly or through neighboring relays. A candidate for bridging disconnected relays must be within two hops of each relay (see secondary relays in Figure 4 (c)). Nodes do not have to store every neighbor's neighbor list. Instead, each node maintains a soft-state list of primary relays within two hops and tags each to indicate if it is reachable. This list is included in *OrgHello* messages. Because primary relays that are two hops away are discovered indirectly, primary relay addresses are tagged with a sequence number (updated by the primary relay upon state changes) to ensure asynchronously received data remains fresh and consistent. Upon receipt of an *OrgHello* message, a node updates an adjacency list to track connectivity between primary relays in the two-hop neighborhood. Secondary relays bridge disconnected primary relays, using an election technique similar to other backbone selection protocols [5]. When a non-relay node discovers two or more primary relays in its two-hop neighborhood that are not connected via the backbone, it sets a random timer to select itself as a secondary relay. The random timer is inversely proportional to a weighted average of the number and proximity of disconnected primary relays and the relay metric of the node and will tend to select the best-connected nodes with the best relay metrics to bridge gaps. The result is a fully connected topology, as shown in three example cases in Figure 4. In the worst case (Figure 4 (c)) two nodes must independently elect themselves as secondary relays to connect a pair of primary relays. ReOrg does not guarantee election of a minimum number of secondary relays, since nodes are only aware of their two-hop neighborhoods. However, because each node only tracks its local neighborhood, storage and communication complexity are low, allowing implementation on resource constrained nodes.

As relay metrics change (e.g., due to greater energy consumption by relays), nodes dynamically update the backbone by changing relay selections using the mechanisms described above, thus spreading the burden of routing over time. When a relay becomes a non-relay, it should inform the routing layer that routes must be repaired, but it may continue to forward data messages for a short time.

4.2 ReOrg Overheads and Tradeoffs

ReOrg is a proactive protocol that introduces a constant communication overhead to maintain the relay backbone. The *OrgHello* send rate (r_H) is a key parameter that may be tuned to trade off overhead versus resilience to change. The time required to detect a change, such as a node failure or energy metric change, and reorganize accordingly is determined by the sum of the soft-state timeout period (typically a multiple of the *OrgHello* interval $1/r_H$) and maximum random selection delay. In many sensor network applications, topology changes due to node failures and new node arrivals are rare, and changes in the relay metric and RF propagation occur relatively slowly (on the order of many minutes, or even hours). Assuming that node positions change infrequently, a low r_H of one message every two minutes trades off a small energy overhead for a reorganization time of up to ten to twelve minutes.

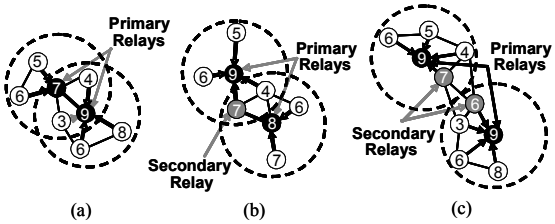


Figure 4: Example ReOrg backbone selection. Cases requiring no secondary relays (a), one secondary relay to directly connect primary relays (b), and two secondary relays to connect primary relays (c). (Relay metric indicated in each node)

ReOrg requires each node to store and exchange a list of primary relays within a two-hop radius. While it is possible for a majority of nodes in a 2-hop radius to be primary relays in extremely sparse topologies (e.g., in a chain of nodes), the fraction drops dramatically as density increases, as node neighborhoods tend to overlap. Assuming uniform density, a node with n 1-hop neighbors may have up to $4n$ neighbors within a 2-hop radius (doubling the radius of a circle results in a 4x increase in area), bounding the primary relay count. Thus, a node with n neighbors requires storage of $O(n)$ records to track primary relays, where each record is a primary relay's address and state. In practice, however, a node in a dense network will often have significantly fewer primary relays in a two hop radius than it has immediate neighbors. In a typical experiment in our 54-node network, an average node had twelve neighbors, but only three to four primary relays in its two-hop neighborhood.

The primary measures of goodness when evaluating a relay backbone protocol are the optimality of the selected backbone (usually the number of nodes and their quality, based on the selection metric) and dynamic characteristics such as the ability to rotate backbone participation for load-sharing. The notion of an "optimal" backbone is application specific and might include election of minimum nodes for connectivity, minimum nodes to maintain a given capacity, or maximum spatial separation for path diversity. This paper focuses on experimental results demonstrating a significant decrease in energy consumption when ReOrg is used to reduce communication frequency between neighbors and the ability of ReOrg to spread the burden of routing across nodes to extend the lifetime of the network.

5. RESYNC: NODE SYNCHRONIZATION

While ReOrg allows a reduction in the number of nodes that participate in routing, unless packet transmission consumes significantly more energy than listening, a MAC protocol that supports sleeping is required to save energy. ReSync is a MAC protocol for multi-hop wireless networks that conserves energy by allowing nodes to sleep most of the time, while maintaining connectivity between neighbors. ReSync performs TDMA-like scheduling rather than using a contention-based approach, as in S-MAC, but ReSync provides distributed TDMA scheduling unlike the 802.11 point coordination function's centralized approach. ReSync was specifically tailored for a class of applications that generate data at a very low rate relative to the available bandwidth, common in the area of large multi-hop wireless sensor networks. ReSync allows these applications to save energy by trading off increased latency.

Scheduling in ReSync is distributed at each node dividing time into recurring epochs. The epoch length is constant across the network. Once each epoch, a node declares in an *intent* message its intent to send or not send a data message at some point in the future. Each node uses a local notion of time to track the expected time at which its neighbors will transmit this declaration. This schedule allows a node to sleep when no transmissions are expected. As shown in Figure 5, each node's epoch provides a different frame of reference for expected *intent* messages. This approach uses only local interaction between neighboring nodes and requires no global clock.

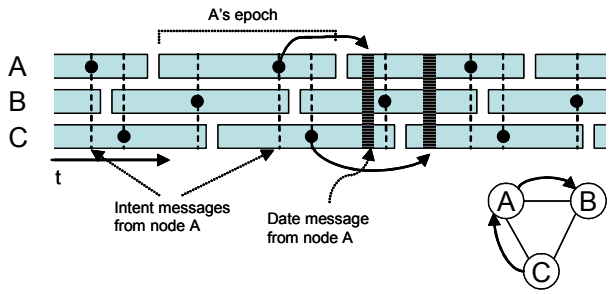


Figure 5: ReSync local *intent* and data message schedules. Each node maintains its own notion of time in terms of epochs and records neighbor intent schedules. Nodes A and C announce the transmission time for data messages, randomly selected to not collide with known transmission schedules.

The ReSync protocol uses two types of messages to synchronize communication between neighbors: small periodic *intent* messages and one-time-scheduled data messages. These mechanisms avoid collisions while maximizing sleep time. Small *intent* messages have a low probability of collision with other messages. In a network with low bandwidth utilization, randomizing send times of larger data messages can avoid persistent hidden terminal collisions.

Participants in ReSync operate in two distinct modes. *Discovery mode* allows a node joining a network to discover the intent schedule of each neighbor and to choose an intent slot. Once the schedule is established, *communication mode* supports message exchange between neighbors.

5.1 ReSync Discovery Mode

Since adding new nodes to a sensor network is a relatively rare event, the primary burden of discovery is placed on nodes that join a network. After powering up, a node initially remains awake for a discovery period of several epochs, promiscuously listening to discover the *intent* message transmission schedules of its neighbors. Figure 6 shows a new node D joining an existing network by discovering its neighbors' intent schedules and subsequently choosing its own *intent* message transmission time such that it does not conflict with existing intent schedules in the neighborhood. Every *N* epochs, each node sends a special *intent* message called a *discovery solicit* message, informing neighbors of a window of time during which the node will listen to discover new neighbors. New nodes that receive this message then send a *newnode* message during this listening period specifying their next *intent* message time. To join an active ReSync network, a node must remain in discovery mode long enough to learn its neighbors' schedules, so that it can subsequently inform each neighbor of its schedule.

When a new node joins a network, it is important that its selected intent schedule not conflict with existing schedules in the neighborhood. One-hop collisions are avoided by scheduling messages at times that are not currently in use by neighboring nodes. Short *intent* messages are used to minimize collisions with two-hop neighbors while providing deterministic scheduling to allow nodes to sleep. Because *intent* messages are small, the probability of *intent* message collision in a two-hop neighborhood is significantly lower than the probability of data message collision. A node can randomly choose the initial time of transmission of its *intent* message, and still have a very low probability of collision with other *intent* messages within a two-hop neighborhood. In the current

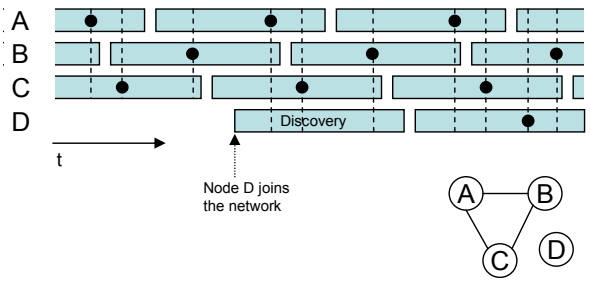


Figure 6: ReSync discovery mode. Node D listens to discover its neighbors' schedules then selects a non-conflicting schedule for its own *intent* messages.

implementation, *intent* messages have a fixed periodicity. However, since any collision that does occur will be persistent, *intent* message times could be varied in a deterministic sequence generated by a pseudo-random function known by each node, similar to a scheduling approach used in SEEDEX [21].

5.2 ReSync Communication Mode

In communication mode, nodes schedule data packet transmission in the context of epochs. During an epoch, each node sends an *intent* message to announce whether or not it has a data packet to send. To transmit a packet a node chooses a random transmission time that does not interfere with locally known transmission times (including *intent* and data messages) of other nodes. The selected transmission time is included in the node's *intent* message and is specified relative to the *intent* message transmission time. When an *intent* message with a relative data message transmission time is received, it is converted into local time. The receiving node later wakes up at the specified time to receive the data message. Figure 5 shows a network of three nodes with node A announcing a data message for transmission to node B. After receiving A's *intent* message, node B knows when to wake to receive the data message.

While the current implementation of ReSync limits the transmission rate of each node to one data message per epoch, the protocol can be extended to send multiple packets per epoch. This extension should increase achievable bandwidth, but it increases the probability of data message collisions. This optimization also reduces the energy usage per data packet by amortizing the cost of an *intent* message transmission over more data packets.

ReSync does not include a mechanism to insure avoidance of hidden terminal data message collisions. An RTS/CTS exchange is not appropriate in a scheme that uses periodic transmissions because neighbors will likely not be awake to hear asynchronous CTS messages. However, ReSync schedules individual data message transmission times randomly. This ensures that two-hop neighborhood data message collisions are not persistent [27].

5.3 ReSync Implementation Challenges

Implementing a TDMA-like protocol such as ReSync is challenging due to timing issues in both the operating system and the hardware.

Several features are required in the underlying operating system. It must be possible to send a packet within a very small margin of error in time, relative to its send deadline. If a deadline is missed (e.g., due to a busy channel), then the message must not be sent at all. This feature is especially important for *intent* messages and ensures the receiver synchronizes with the sender within a small

margin of error. A second requirement is the ability to accurately timestamp received packets [9]. The delivery timestamp is used to calculate the delivery time of the next *intent* message or a data message. To increase accuracy, timestamps are calculated as soon as the preamble of a packet is received, at the encoding/decoding layer.

Hardware limitations also affect the performance of the ReSync implementation. Despite the high accuracy of crystal oscillator used in the current mote design, there is a significant amount of clock skew between nodes. In TDMA-like protocols, clock skew can result in two problems: loss of synchronization with neighbors and a tendency to drift into a neighbor's scheduled slot over time. ReSync avoids the first problem using guard bands and by resynchronizing on each received message. A guard band provides a buffer of time around the expected receive time of a message, allowing a message to be received despite clock drift. It is also possible for the intent slot of two nodes to drift over time and become coincident. To avoid this problem, nodes can use different intent periods, or the intent schedule can be randomized with a known pattern.

6. PROTOCOL EVALUATION

We have evaluated the ReOrg and ReSync protocols in a 54-node experimental testbed. The testbed nodes are a variant of the rene mote [12] wireless sensor platform, originally developed at the University of California at Berkeley. The core components of the mote are an Atmel® ATmega323L AVR® microcontroller with 32 KB of flash, 2 KB of RAM, and an RFM TR1000 916 MHz radio transceiver. Protocols and applications were implemented using TinyOS [12], an event-driven operating system designed to fit within the minimal resources of mote hardware.

The radio provides on-off keying modulation and a raw bit rate of 10 Kbps. SEC/DED (single error correcting, double error detecting) baseband encoding is implemented in software, for a maximum theoretical channel capacity of about 568 Bps. Since each packet is 37 bytes long, the channel capacity has a theoretical limit of 15 packets/sec. The default TinyOS CSMA/CA MAC is used for comparison with ReSync.

In all experiments, nodes were placed in a grid (Figure 7). Nodes spacing was approximately 3.5 inches, and the transmit power was tuned for an average packet reception distance of eight inches. This topology results in a network that is approximately two to three hops wide and four hops long. While a real sensor network would not typically be deployed in such a uniform fashion, a grid topology was chosen to allow a repeatable evaluation of ReOrg and ReSync performance. The results obtained from the grid layout are not overly dependent on a specific application topology.

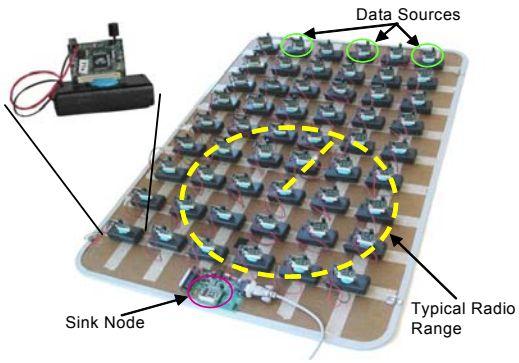


Figure 7: Topology map for the testbed used in the experimental evaluation of ReOrg and ReSync. The density of the 54 node network is approximately 16 nodes/sq. ft.

While ReOrg and ReSync support any-to-any communication, we chose to evaluate the protocols with a many-to-one communication model, as is used in the conference room monitoring application. Three nodes located at one end of the testbed inject packets at a constant rate. A gateway node is centered near the opposite edge of the network (approximately four radio hops from the source nodes) to receive data packets and collect statistics. The remaining nodes participate in ad hoc networking to forward data packets across multiple hops. This deployment approximates a dense sensor network application where a central server gathers data (e.g., local temperature readings) from individual sensors. To collect comparable statistics across experiments, the three data sources were placed at approximately the same distance from the gateway node and therefore exhibit similar performance characteristics.

Experiments were conducted using data rates ranging from zero to three packets per minute, as might be required to support typical sensor network applications (e.g., building occupancy monitoring or temperature sensing). Sensor values in such applications tend to change slowly or infrequently. For example, our maximum data rate is equivalent to 45 sensors each sending one packet every five minutes. Such a low data rate can also help the network to last longer. ReSync was configured with an epoch length of 10 seconds, allowing each node to transmit up to six packets per minute. This value was chosen to allow experiments with a variety of data rates. Note, however, that this is a relatively high duty-cycle for many low data rate applications. In a real-world network that must survive on a single set of batteries for a year or more, an even larger epoch length could be used to further reduce the duty cycle. Finally, ReOrg was configured with an *OrgHello* interval of two minutes, with the assumption that topology changes will be driven primarily by slow-changing energy values rather than mobility or frequent additions or removal of nodes. The energy consumption overhead of ReSync and ReSync+ReOrg is captured in the zero data rate measurements since no data traffic is included in these experiments.

To evaluate the performance of our protocols, a set of experiments was performed using various protocol combinations (Table 1). The mote's 32 KB code space was a limiting factor in our testbed experiments (Table 2). Due to code size limitations and because our study was not focused on routing, we use a simple routing protocol (flood) on top of our protocols to collect data from sensors.

The following sections evaluate the energy savings achieved by the ReOrg and ReSync protocols, the benefit of tightly integrating these protocols, the importance of normalizing for packet loss, the effect of ReOrg and ReSync on network lifetime, and the ability of these protocols to leverage node heterogeneity. Each experiment was run three times, and results include 90% confidence intervals.

6.1 Energy Savings Through Sleeping

Figure 8 illustrates the raw energy consumption of flooding with no energy saving protocols, ReSync alone, and ReOrg running on top of ReSync with two example energy models. Our results are based on mathematical models of energy consumption described in detail in Section 6.1.1. The average energy consumed per node during a 60 minute experiment is reported for each protocol combination. With the first energy model (EM1), based on current mote hardware, ReSync reduces average energy consumption by 63% at the

Table 1: Protocol combinations used in experiments.

Label	Description
Flood	Flooding with no energy saving protocols
Flood+Sync	Flooding with ReSync
Flood+Sync+Org	Flooding with ReOrg and ReSync

Table 2: Code size (out of 32 KB) for various components used in the implementation.

Component	Code Size (bytes)
ReSync protocol	8400
Network programming radio stack	6400
ReOrg protocol	5700
TinyOS	5400
Experiments	3000
Utilities	2200
Flood routing protocol	700
Total:	31800

lowest data rate and 54% at the highest data rate. The addition of ReOrg at low data rates reduces the energy savings to as little as 61%, due to the cost of periodic *OrgHello* messages. At higher data rates, *OrgHello* overhead is amortized over a large number of data messages, but the added benefit is small, up to only 57%.

6.1.1 Energy Models

The use of energy models in the experimental evaluation of a wireless network may seem non-intuitive, but gathering energy characteristics from a testbed of tiny sensor devices is a challenge. While it is possible to use lab equipment such as voltmeters and oscilloscopes to precisely measure the energy consumption on a single sensor node, it is impractical to use this approach to monitor runtime energy consumption performance on each node in a large-scale sensor network. Energy monitoring hardware, commonly available for reporting remaining battery capacity in mobile computing devices such as laptops, handhelds, and cell phones, is primarily tuned for measuring current flows of hundreds of milliamps or more. They are not effective at measuring very small currents in the microamp to milliamp range common in sensor network devices.

For these reasons, an energy model based on the time a node spends in four key operational modes combined with typical current consumption in each of these modes is used to measure energy consumption. During an experiment, each node tracks the time spent transmitting (*TX*), receiving (*RX*), computing without communicating (*CPU*), and sleeping in a low power state (*Save*). We assume that energy consumption for listening is no greater than receiving and include both in RX time. While these values could be mapped to any hardware energy model, we applied two energy models based on mote sensor node hardware, listed in Table 3, to report the estimated energy consumed during the course of each experiment, as shown in equation 1, where T is the experiment time duration and n is the total number of operational modes in the energy model.

$$\text{Energy}_{\text{raw}} = \left[\sum_{\text{mode}=1}^{n} (\%\,\text{time}_{\text{mode}}) \times (\text{Power}_{\text{mode}}) \right] \times \text{T} \qquad (1)$$

Neither of these energy models includes energy consumed by sensors which may be attached to the wireless nodes, as this is a very application-specific parameter. It is worth noting that sleeping saves energy for some types of sensors, but not all sensors. Some sensors, such as sophisticated weather monitoring devices, require more energy than the sensor node itself, trivializing the benefit of energy saving protocols. However, many passive sensor devices have a very short warm-up time or require a negligible amount of power relative to the rest of the system, such as thermistors, passive infra-red sensors, and light sensors [20].

Table 3: Energy models based on testbed hardware. TX and RX modes include energy consumption of the radio and CPU combined. Save mode is the lowest power sleep mode supported by the radio and CPU. Energy consumption for listening is assumed to be identical to RX.

Modes:	TX	RX	Save	CPU
Energy Model #1 (EM1): *Current Mote Hardware*	17 mA	9.5 mA	2.5 mA	5 mA
Energy Model #2 (EM2): *Improved Mote Hardware*	17 mA	9.5 mA	0.006 mA	5 mA

6.1.2 Benefits of improved hardware

Improvements in mote hardware, represented by energy model 2 (EM2) can allow a significantly greater energy savings. While EM1 represents the current testbed node design, EM2 includes important energy-friendly enhancements to the current hardware design. As shown in Figure 8, with EM2 the energy savings of ReSync+ReOrg jumps to between 83% at low data rates and 78% at high data rates.

The only difference between the two models is the energy consumption in *Save* mode. Because the current mote hardware uses a crystal oscillator, a long recovery period is required to resume from the lowest power sleep mode. Due to this limitation, EM1 assumes that the processor is not put in the lowest power sleep mode to avoid turning off the oscillator, increasing the energy consumption of the *Save* mode. EM2 assumes the use of an independently powered crystal, allowing a very low energy state without requiring a long wakeup period. As demonstrated by our results, reducing the power consumption of nodes while they are not communicating dramatically improves the energy savings achievable by these protocols.

6.2 Energy Benefits from Tight Integration

In the results from Section 6.1, running ReOrg as an independent layer above ReSync provides little, if any, additional benefit. Figure 9 demonstrates that tightly integrating the two protocols reduces energy consumption by 92% at low data rates and 82% at high data

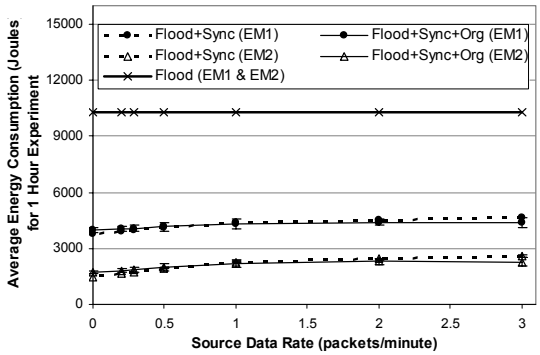


Figure 8: Average energy consumption of Flood, Flood+Sync, and loosely integrated Flood+Sync+Org at various data rates. The largest confidence interval in this graph is 178 Joules.

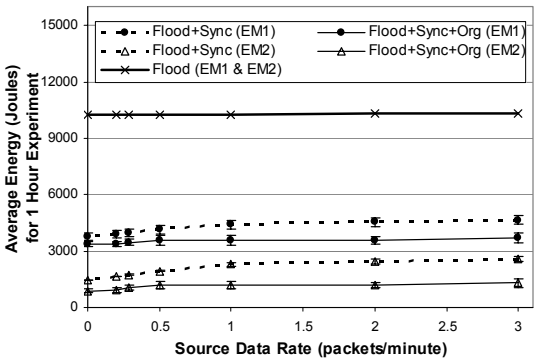


Figure 9: Average energy consumption of Flood, Flood+Sync, and tightly integrated Flood+Sync+Org at various data rates. Results are identical to Figure 8 except for Flood+Sync+Org. The largest confidence interval is 260 Joules.

rates when using energy model 2.

Although a stand-alone implementation of ReOrg reduces the number of transmissions in the network, in low-power wireless devices, simply listening for radio broadcasts from neighbors can be nearly as expensive as transmitting [10] [20] (energy consumption for communication modes on Berkeley mote hardware are provided in Table 3). Thus, reducing the frequency with which nodes listen to individual neighbors can result in significant additional energy savings. Through tight integration, ReOrg can instruct ReSync about links that are not on the selected graph, and thus will not be used for communication. This approach allows ReSync to save significantly more energy by listening to fewer neighbors. Figure 9 shows the benefits of tightly integrating the backbone and MAC layers.

6.2.1 Adaptive Listening Frequency

OrgHello messages are transmitted by each node every *n*th send interval, called the *OrgHello* interval. At a minimum, every node, regardless of whether or not it is a relay, must listen for an *intent* and subsequent data message from each neighbor once every *OrgHello* interval to maintain the backbone. In addition, nodes along the backbone must listen to specific neighbors more frequently to route data messages. In particular, primary relays must wake at every send interval to listen for *intent* messages from each relay neighbor (primary and secondary), to forward data along the backbone, and they must listen to receive packets from all neighboring non-relay nodes that selected them (Section 4.1). For example, in Figure 10 primary relay node 3 must listen to relay nodes 8 and 7 and to non-relay neighbors 1, 2, and 4 at every interval. Node 3 need not listen to nodes 5 and 6, as they did not elect node 3 as a primary relay. Secondary relays, on the other hand, don't support any non-relay nodes, and thus only listen to neighboring relays at every send interval. For example, secondary relay node 7 only listens to relay nodes 3 and 8 at every interval. Non-relays must only wake up for the primary relay they selected at every send interval to receive messages from the backbone. For example, node 9 only listens to primary relay node 8 at every interval.

These integration rules allow significant savings by promoting selective listening between neighbors. In a typical run of our 54 node testbed experiment, ReOrg selected seven primary relays and 12 secondary relays. On average, each node had 12 neighbors; four were relays. Each primary relay was selected by an average of eight non-relays. In this scenario, with a *OrgHello* message listen interval of one out of four epochs, integration allows an average listening time reduction (over non-integrated ReSync) of more than 50% for secondary relays and 70% for non-relays. An average node can thus sleep 56% of the time it would have otherwise been listening.

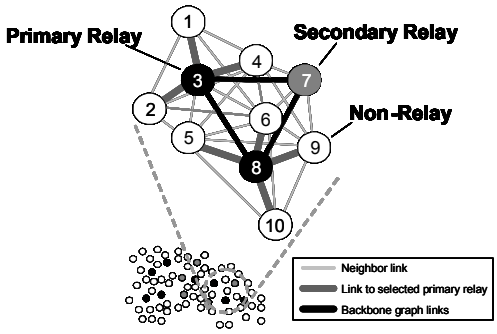


Figure 10: Selective listening on a ReOrg backbone. Every node listens to each neighbor once every OrgHello interval. Dark lines signify neighbors listened to every interval.

6.2.2 Tradeoffs in Adaptive Listening

The fraction of listen intervals required for *OrgHello* messages limits energy savings achievable by integrating ReOrg and ReSync. For example, if the fraction is reduced from 1/4 to 1/8 (e.g., by increasing ReSync's maximum send-rate or decreasing the *OrgHello* send interval), secondary relays would experience a listening time reduction of 58% and non-relays 80%, for an overall reduction of 65% across the network (as compared with non-integrated ReSync). Thus, overall energy consumption would be reduced.

6.3 Normalizing for Packet Loss

While these results appear very promising, it is important to consider the complete picture of energy consumption in order to evaluate the benefit of these protocols. Figure 11 shows the mean end-to-end data delivery rate recorded from each experiment, based on monotonically increasing sequence numbers included in data packets. Multi-hop wireless sensor networks are known to produce significant packet loss [31]. The results for flood alone indicate a data packet loss of 10-18%, even when no energy saving protocols are used. In many cases the use of ReSync and ReOrg drive the delivery ratios even lower, especially at higher data rates.

The vertical line in Figure 11 represents the theoretical capacity

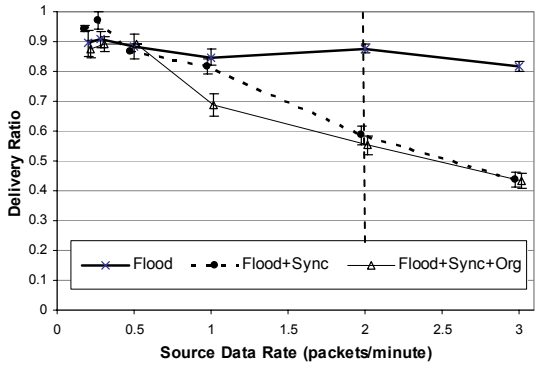


Figure 11: Delivery ratio as a function of transmission rate for Flood, Flood+Sync, tightly integrated Flood+Sync+Org.

bound supported by ReSync when flooding. As described in Section 5.2, each node can transmit at most one data packet per 10 second epoch, resulting in a maximum send rate of six packets per minute per node. With each of the three source nodes generating two packets per minute, each node must relay a total of six packets per minute (the capacity bound). Data rates above this threshold will result in a significant number of dropped packets. Thus, performance degrades near this threshold due to bandwidth saturation.

Note that ReSync delivery ratios are somewhat lower at most data rates. We believe this is due to a limit to the number of neighbors that ReSync can track in the current implementation. We verified this hypothesis by running ReSync in a network with lower density. It is also worth noting that the use of ReOrg combined with ReSync also decreases delivery ratios, primarily due to a reduction of available redundant paths in the backbone in comparison to the non-ReOrg network where every node relays messages.

Many studies evaluating the energy performance of ad hoc wireless protocols report the total energy consumed by the network [5] [15] [28]. However, raw energy consumption does not always provide a complete picture. In particular, many protocols trade off characteristics such as reliability or throughput to conserve energy. Since downstream nodes will not expend energy forwarding a packet once it has been lost, raw energy metrics can be artificially improved as a result of increased loss rates. To eliminate the contribution of end-to-end packet loss, the results must be normalized. One metric that more fairly measures energy consumption when trading off data loss is goodput energy: the energy consumed per packet received, relative to the total number of end-to-end packets transmitted during the experiment. This is equivalent to dividing the energy by the end-to-end delivery ratio, as shown in equation 2.

$$\text{Energy}_{\text{normalize}_2} = \text{Energy}_{\text{raw}} \times \frac{1}{\text{Delivery Ratio}} \qquad (2)$$

Figure 12 presents goodput energy normalized from the data in Figure 9. As a protocol's delivery ratio decreases, its goodput energy consumption increases. When energy metrics are adjusted to account for packet loss, the benefits of the energy saving protocols may not be as compelling. However, despite packet loss, ReSync alone and tightly integrated ReOrg and ReSync still reduce energy used by the network. ReSync saves between 53-85% and tightly integrated ReOrg with ReSync saves between 76-91% with the normalized metric for energy model 2.

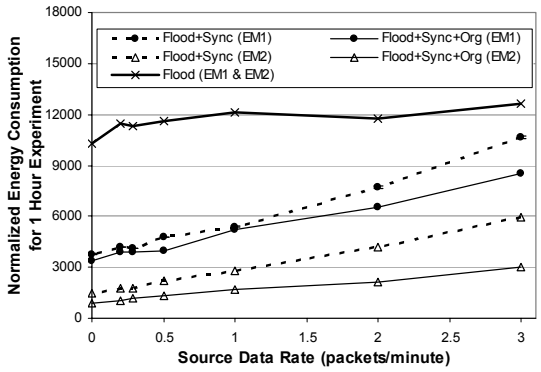


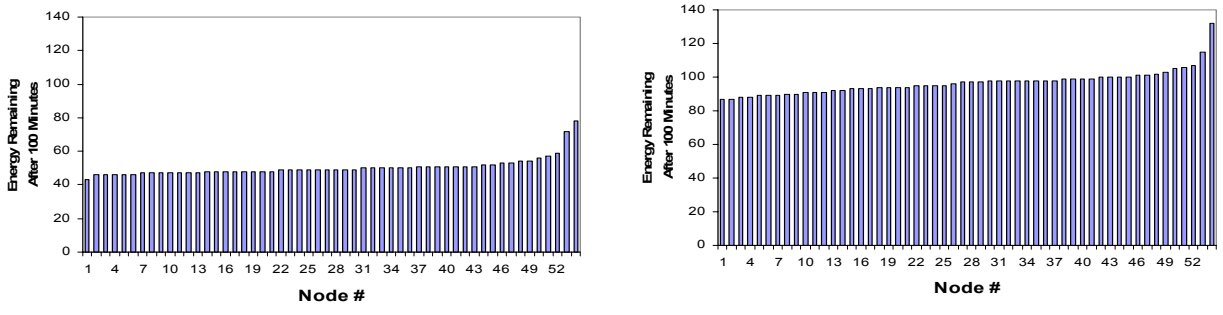
Figure 12: Average energy consumption normalized for delivery ratio of Flood, Flood+Sync, and tightly integrated Flood+Sync+Org at various data rates.

6.4 Load Balancing and Network Lifetime

While we have shown that ReOrg and ReSync can be used to reduce the total amount of energy consumed by a network to perform a particular task, these results do not necessarily demonstrate that these protocols increase the lifetime of a sensor network. In particular, unless energy is consumed evenly throughout the network, some nodes could be drained very rapidly to allow other nodes to use very little energy. This section demonstrates that our protocols can increase network lifetime by 5-10 times, corresponding to the previously reported 82-92% reduction in energy consumption.

To measure the impact of our energy conserving protocols on network lifetime, we initialized the energy monitoring component on each node with the same energy capacity and then ran experiments long enough to completely deplete the energy from network nodes. Note that we did not initialize nodes with a typical AA battery capacity of 4000 mA-hours because the average energy consumption values shown in Figure 9 would require experiments to run for several months before batteries would be drained. To allow multiple experiments to be run in a reasonable amount of time, we instead chose an arbitrary initial battery capacity of 200 mA-minutes for each node. These experiments were run using energy model 2.

Figure 13 presents histograms of remaining energy capacity for individual nodes at the midpoint of a lifetime experiment. After 100 minutes, (a) and (b) show each node's remaining energy with ReSync alone and with ReOrg tightly integrated with ReSync, respec-



(a) (b)

Figure 13: Histograms of energy drain of individual nodes using (a) Flood+Sync and (b) tightly integrated Flood+Sync+Org, recorded at the midpoint of an experiment.

tively. We would expect that with ReSync alone, the energy drain across all nodes would be approximately flat, since the Flood protocol requires each node to transmit each packet exactly once. The outlier nodes in Figure 13 (a) are due to the edge effect of the rectangular topology. Similarity in the shapes of the histograms in Figure 13 (a) and Figure 13 (b) suggest that ReOrg balances energy consumption among nodes, rather than sacrificing a few select nodes to decrease average energy consumption across the network. The distribution of remaining energy is approximately the same whether or not ReOrg is used, yet nodes in the ReOrg network have roughly twice as much energy remaining after 100 minutes in comparison to nodes in the network with only ReSync. In addition, the relatively flat energy distribution indicates that lifetime experiments based on 200 mA-minutes initial energy capacity were sufficiently long to allow ReOrg to spread energy burden across network nodes when using a hello interval of two minutes (see Section 4.2).

Figure 14 shows the measured lifetime of the network when running each protocol combination, with lifetime defined as the time until the first network node is drained. By allowing nodes to sleep much of the time, ReSync increases network lifetime by 3-6 times, depending on traffic load. Since ReOrg spreads the routing burden, it extends network lifetime even further when tightly integrated with ReSync, resulting in an overall lifetime improvement of 5-10 times.

Recall that the results from Figure 9 showed an average decrease in energy consumption of 82-92% when using ReOrg tightly integrated with ReSync, which is consistent with a lifetime increase of 5-10X. Thus, average energy usage is not simply improved by sacrificing a subset of nodes. The overhead of running synchronization and organization in the network is low, allowing substantial lifetime improvements even at low data rates. We achieved similar results with lifetime defined as 85% node survival (Figure 15), demonstrating that our results are relatively independent of how "network lifetime" is defined.

While network lifetime results presented in this paper are based on node battery capacities of 200 mA-minutes, these results can be extrapolated to nodes powered by two AA batteries (4000 mA-hours). In this case, tightly integrated ReOrg and ReSync would increase network lifetime from 17 days to 94-170 days. We are in the process of validating network lifetime when using AA batteries.

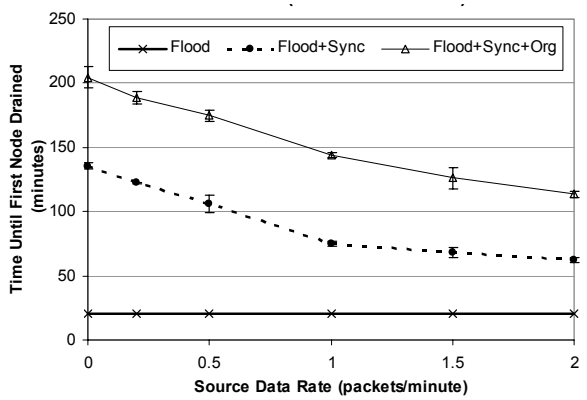


Figure 14: Network lifetime (first node drained), as a function of data rate for Flood, Flood+Sync, and tightly integrated Flood+Sync+Org.

6.5 Heterogeneity and Network Lifetime

Without modification ReOrg can also be used to leverage heterogeneity in an ad hoc network. When a subset of nodes in a network have a higher capacity battery, or are wall powered, ReOrg will regularly elect those nodes as primary relays, allowing them to perform a greater portion of the network's packet forwarding burden.

Figure 16 demonstrates the effect of a varying number of evenly distributed wall-powered nodes on the lifetime of a 54-node network with a one packet per minute data rate. As expected, in a network without ReOrg, the lifetime remains flat as wall-powered nodes are added since ReSync alone does not leverage heterogeneity. With ReOrg, lifetime is increased even when adding only a small number of wall-powered nodes. However, as the number of wall-powered nodes increases, network lifetime does not continue to improve forever. It is bounded by the overhead of ReOrg and ReSync and occurs when no data is transmitted. Thus, the upper bound shown in Figure 16 is 218 minutes, taken from Figure 15. The lower bound of 156 minutes occurs when no wall-powered nodes are used, previously reported in Figure 15.

Increasing the data rate decreases the network lifetime when no wall-powered nodes are used (corresponding to the lower bound). In this case, wall-powered nodes provide a greater potential for benefit. Note that the upper bound of achievable energy savings when adding wall-powered nodes can be increased by lengthening the ReSync epoch or reducing the ReOrg *OrgHello* interval, thus reducing the baseline duty cycle of communication. However, such changes limit the maximum data rate supported by the network.

6.6 Comparison to Prior Simulation Results

Span, which has similar goals to ReOrg, has been evaluated using simulations. Because Span has similar message and computation overhead to ReOrg, we expect that Span would likely exhibit similar behavior to ReOrg in an experimental testbed evaluation. The Span evaluation paper [5] argues that a hexagonal grid approximates an optimal backbone election, and simulation results showed that backbones elected by Span in a random topology were typically around three times this optimal density. In our 54-node testbed (Figure 7), with a density of roughly 16 nodes/ft^2, the hexagonal approximation would result in approximately 2 relays/ft^2. In a typical 54-node experiment, ReOrg elected 19 relays, or approximately 6 relays/ft^2, which is about three times the number of relays in the

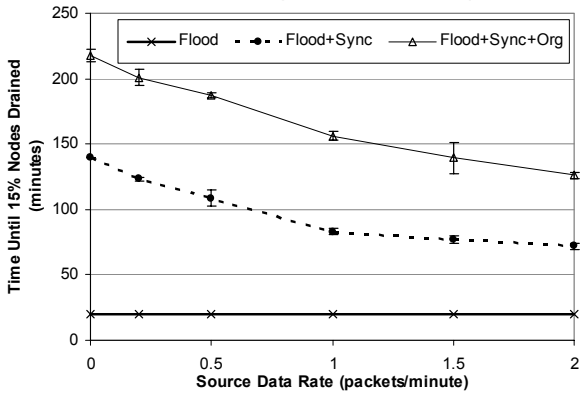


Figure 15: Network lifetime (15% nodes drained), as a function of data rate for Flood, Flood+Sync, and tightly integrated Flood+Sync+Org.

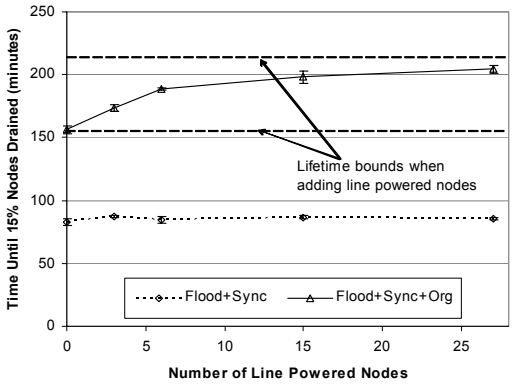


Figure 16: Network lifetime at 1 packet/min with a varying number of wall-powered nodes for Flood+Sync and tightly integrated Flood+Sync+Org.

hexagonal case and in line with the Span simulation results. While the paper demonstrates a decrease in packet loss in comparison to 802.11 when using Span with geographic routing at relatively low data rates and high mobility, it also shows that delivery ratios with Span are lower than those of 802.11 at high data rates. As our experimental evaluation was focused on sensor networks, we did not evaluate our protocols with mobility. However, the lower delivery ratios of Span at high data rates are similar to the decreased delivery ratio observed in Figure 11 when using ReOrg.

7. FUTURE WORK

Section 6 evaluates the benefit of integrating a topology control protocol with an energy-saving MAC protocol, allowing nodes to listen to a select subset of neighbors and sleep otherwise. We are currently evaluating the benefit of further integrating ReSync with other ad hoc routing protocols. By listening only to neighbors on active routing paths, even greater energy savings may be possible. We will also evaluate the benefit of integrating ReOrg with ad hoc routing, reducing the impact of using flood for route maintenance.

In Section 4, we argued that it is beneficial to separate small *intent* messages used to schedule data messages from the data messages themselves. In future experiments, we will modify ReSync to optionally send full data messages using the technique currently used to schedule *intent* messages, and compare the results with the current ReSync implementation to explicitly measure the benefit of small *intent* messages.

Finally, this paper provided a brief comparison between our experimental results and previously published simulation results. As follow-on work, we would like to implement related protocols such as Span in our testbed and evaluate the protocols with identical network configuration and routing protocols to provide a more accurate comparison.

8. CONCLUSIONS

The evaluation of wireless networking protocols on real hardware using experimental measurements provides performance insights that are often overlooked in simulations. This is particularly true for sensor networks, where packet delivery ratios in real-world networks are often significantly lower than what is observed in simulations. This paper presented two energy saving protocols that work together to allow sensor network nodes to conserve energy by sleeping. Our evaluation of these protocols provides an example execution of a set of techniques for estimating node energy consumption, weighing the impact of different hardware design choices, and considering the impact of packet loss. This evaluation also serves to demonstrate the benefit of these protocols in reducing energy consumption and extending the lifetime of sensor networks. We also show that tight integration of topology control and energy-saving MAC protocols magnifies these energy conservation effects. Finally, we demonstrate the use of our protocols to allow a small number of wall-powered nodes to greatly increase the lifetime of a battery-powered network. In future we plan to expand our investigation into the integration of ReSync and ReOrg with an ad-hoc routing protocol and to evaluate the impact of ReSync's short *intent* messages on its overall performance.

9. REFERENCES

[1] Baker D.J., and Ephremides, A. The architectural organization of a mobile radio network via a distributed algorithm. IEEE Trans. on Comm., 29, 11 (Nov. 1981).

[2] Barrett, C., Drozda, M., Marathe, A., and Marathe, M. Characterizing the interaction between routing and MAC protocols in ad-hoc networks in Proc. ACM MOBICOM 2002 (Atlanta GA, June 2002).

[3] Cerpa, A., and Estrin, D. ASCENT: Adaptive self-configuring sensor networks topologies in Proc. INFOCOM '02 (NY, NY, June 2002).

[4] Chang, J., and Tassiulas, L. Energy conserving routing in wireless ad-hoc networks in Proc. IEEE INFOCOM 2000 (Israel, Mar. 2000).

[5] Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks in Proc. MOBICOM '01 (Rome, Italy, July 2001).

[6] Clark, B.N., Colbourn, C.J., and Johnson, D.S. Unit disk graphs. Discrete Mathematics, 86 (1990).

[7] Conner, W.S., Krishnamurthy, L., and Want, R. Making everyday life easier using dense sensor networks in Proc. of ACM Ubicomp 2001 (Atlanta GA, Oct. 2001).

[8] Doshi, S., and Brown, T. Minimum energy routing schemes for a wireless ad hoc network in Proc. IEEE INFOCOM 2002 (New York NY, June 2002).

[9] Elson, J., Girod, L., and Estrin, D. Fine-grained network time synchronization using reference broadcasts in Proc. OSDI 2002 (Boston MA, Dec. 2002).

[10] Feeney, L.M. An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks in ACM Mobile Networks and Applications, 6, 3 (June 2001).

[11] Heinzelman, W., Chanddrakasan, A., and Balakrishnan, H. Energy-efficient communication protocols for wireless microsensor networks in Proc. Hawaii International Conf. on Systems Sciences (Jan. 2000).

[12] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. System architecture directions for networked sensors in Proc. ACM ASPLOS 2000 (Cambridge MA, Nov. 2000).

[13] Intanagonwiwat, C., Govindan, R., and Estrin, D. Directed diffusion: A scalable and robust communication paradigm for sensor networks In Proc. ACM MOBICOM 2000 in (Boston MA, Aug. 2000).

[14] LAN MAN Standards Committee of the IEEE Computer Society. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, ANSI/IEEE Std. 802.11 (1999).

[15] Lindgren, A., and Schelen, O. Infrastructured ad hoc networks in Proc. International Workshop on Ad hoc Networking 2002 (Vancouver, BC, Canada, Aug. 2002).

[16] Mainwaring A., Polastre, J., Szewczyk, R., and Culler D. Wireless sensor networks for habitat monitoring in Proc. ACM Workshop on Sensor Networks and Applications (Atlanta GA, Sept. 2002).

[17] Min, R., Bhardwaj, M., Ickes, N., Wang, A., and Chandrakasan, A. The hardware and the network: Total-system strategies for power aware wireless microsensors in Proc. 2002 IEEE CAS Workshop (Sept. 2002).

[18] Ni, S., Tseng, Y., Chen, Y., and Sheu, J. The broadcast storm problem in a mobile ad hoc network in Proc. MobiCom '99 (Seattle WA, Aug. 1999).

[19] Perkins, C.E., Royer, E.M., Das, S.R. Ad-hoc on-demand distance vector (AODV) routing. IETF draft (Mar. 2002).

[20] Raghunathan, V., Schurgers, C., Park, S., and Srivastava, M.B. Energy aware wireless microsensor networks in IEEE Signal Processing Magazine, 19, 2 (Mar. 2002).

[21] Rozovsky, R., and Kumar, P. SEEDEX: A MAC protocol for ad hoc networks in Proc. ACM MobiHoc 2001 (Long Beach CA, Oct. 2001).

[22] Royer, E.M., Lee, S., and Perkins, C.E. The effects of MAC protocols on ad hoc network communications in Proc. IEEE Wireless Comm. and Networking Conference (Chicago IL, Sept. 2000).

[23] Schurgers, C., Tsiatsis, C., and Srivastava, M. STEM: Topology management for energy efficient sensor networks in Proc. 2002 IEEE Aerospace Conference (Mar. 2002).

[24] Singh, S., Woo, M., Raghavendra, C.S. Power-aware routing in mobile ad hoc networks in Proc. ACM MOBICOM 1998 (Dallas TX, Oct. 1998).

[25] Sohrabi, K., Gao, J., Ailawadhi, V., and Pottie, G. Protocols for self-organization of a wireless sensor network in IEEE Personal Communications (Oct. 2000).

[26] Trujillo, S., CEO Graviton. UCSD Connect SENSORNET Conference Keynote (April 29, 2002).

[27] Woo, A., and Culler, D. A transmission control scheme for media access in sensor networks in Proc. MOBICOM '01 (Rome, Italy, July 2001).

[28] Wu, J., Dai, F., Gao, M., and Stojmenovic, I. On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks in Journal of Comm. and Networks, 4, 1 (Mar. 2002).

[29] Xu, Y., Bien, S., Mori, Y., Heidemann, J., and Estrin, D. Topology control protocols to conserve energy in wireless ad hoc networks. Center for Embedded Networked Computing Technical Report 6, University of California, Los Angeles (Jan. 2003).

[30] Xu, Y., Heidemann, J., and Estrin, D. Geography informed energy conservation for ad hoc routing in Proc. ACM MOBICOM 2001 (Rome, Italy, July 2001).

[31] Yarvis, M.D., Conner, W.S., Krishnamurthy, L., Chhabra, J., Elliot, B., and Mainwaring, A. Real-world experiences with an interactive ad hoc sensor network in Proc. International Workshop on Ad hoc Networking (Vancouver, BC, Canada, Aug. 2002).

[32] Ye, W., Heidemann, J., and Estrin, D. An energy-efficient MAC protocol for wireless sensor networks in Proc. IEEE INFOCOM 2002 (New York NY, June 2002).

[33] Yu, Y., Govindan, R., and Estrin, D. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical Report TR-01-0023, UCLA, Computer Science Department (2001).